Information Sciences 625 (2023) 417–429

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Reinforcement learning for control design of uncertain polytopic systems

Pedro M. Oliveira [a], Jonathan M. Palma [b], Erivelton G. Nepomuceno [c], Márcio J. Lacerda [a,*]

[a] *Federal University of São João del-Rei, Department of Electrical Engineering, Brazil*
[b] *Faculty of Engineering, Universidad de Talca, Curicó, Chile*
[c] *Centre for Ocean Energy Research, Department of Electronic Engineering, Maynooth University, Ireland*

A R T I C L E I N F O

Article history:
Received 20 December 2021
Received in revised form 9 May 2022
Accepted 3 January 2023
Available online 6 January 2023

Keywords:
Reinforcement learning
SARSA
Robust state-feedback
Static output-feedback
Polytopic uncertainty
Pole placement

A B S T R A C T

This work is concerned with the design of state-feedback, and static output-feedback controllers for uncertain discrete-time systems. The reinforcement learning (RL) method is employed and the controller to be designed is considered as an agent changing the behavior of the plant, which is the environment. A State-Action-Reward-State-Action (SARSA) algorithm is developed to achieve this goal. This is an open problem, as this offline design through the usage of RL is an approach not so well explored in the literature. The gain matrices are used directly as design variables in the SARSA algorithm, and a time-varying incremental step is employed. The method uses a grid in the uncertain parameters to place the poles of the closed-loop system in a disk on the complex plane. In addition, a stability test based on the Lyapunov theory is performed to provide a hard stability certificate for the closed-loop system. Numerical experiments from the literature are used to illustrate the efficacy of the method, through the use of benchmark examples and exhaustive testing.

1. Introduction

Reinforcement Learning (RL) is a subfield of machine learning [19], and it has been widely employed in several areas, mainly due to its learning abilities, that can provide solutions for problems with little information about the environment [33]. One may cite the salesman problem [26,27], sequential ordering problem [25], multidimensional knapsack problem [6], power systems transient stability [9], applications in communications and networking [21], and autonomous driving [16]. There are many algorithms and methods to solve RL problems, with each being more appropriate to certain scenarios. Methods based on Monte-Carlo [15], Dynamic Programming [4], Temporal Difference [33], as well as the Actor-Critic method [14] can be brought forth. It is also worth mentioning Deep Reinforcement Learning (DRL) [3], which fuse concepts from RL with deep neural networks, allowing to tackle highly complex problems with an elevated number of states.

Reinforcement learning for control design has received significant attention in the last years, among others one may find works dealing with tracking control for time-delayed non-linear systems [22,24] where a distributed control strategy was employed in a model-free scenario with stability guarantees for a class of interconnected linear subsystems. Instances where metaheuristic algorithms were employed in combination with RL can also be found, with methods such as Gravitational

* Corresponding author.
E-mail addresses: pe-droliveira@hotmail.com (P.M. Oliveira), jonathan.palma@utalca.cl (J.M. Palma), erivelton.nepomuceno@mu.ie (E.G. Nepomuceno), lacerda@ufsj.edu.br (M.J. Lacerda).

https://doi.org/10.1016/j.ins.2023.01.042
0020-0255/© 2023 Elsevier Inc. All rights reserved.

Search Algorithm (GSA) [37] and Grey Wolf Optimizer Algorithm (GWO) [36] being used to train the Neural Networks utilized. In [17], the $\mathscr{H}_\infty$ performance was used within the reinforcement learning method to solve the game algebraic Riccati equation for discrete-time precisely known systems.

Different from existing works using reinforcement learning for control systems, the class of uncertain linear models is considered in this paper. It is well known that the design of controllers for uncertain systems is a very challenging problem. One may find results based on the use of Riccati equations [12,13], and also methods that rely on the Lyapunov theory, deriving Linear Matrix Inequalities (LMIs) that provide sufficient conditions for the design of state and static output-feedback controllers [23,11]. Recently, approaches based on an iterative method have emerged, for instance [28] which makes use of past measured outputs to design non-minimal controllers. Furthermore, the approach in [30] applies an iterative framework to reduce conservatism in both state-feedback and output-feedback design problems. However, these methods' design techniques are also based on the solution of LMIs. It is a consensus that Lyapunov-based methods for control design of uncertain systems have had increasing difficulty in reducing conservatism [31]. In contrast, the stability certificates for uncertain systems via LMIs and Lyapunov theory have provided necessary and sufficient conditions to guarantee the stability of the uncertain systems [7,29].

The main drawback of the use of Lyapunov-based methods for control design is to obtain convex formulations that can be solved by the available semi-definite programming software. The methods rely on structural constraints, polynomial degrees, and specific structures imposed on the Lyapunov function, which can lead to the introduction of conservatism. In [5] the authors have proposed a method based on the use of genetic algorithms and in the solution of LMIs to solve the pole placement problem by using static output-feedback controllers. In [35], a set of LMIs is presented to design a full order dynamic output-feedback controller which can assign the poles of the closed-loop system and guarantees an upper bound to the output covariance of the controlled system.

In this paper, we are addressing both the state-feedback and the static output-feedback design problems. The state feedback controller assumes that all the states from the system are accessible and can be utilized as inputs on the controller. However, this full readability is not always possible in practical implementations. In contrast, in the output-feedback controller, the output of the system is utilized as input to the controller. It is more applicable since the output is generally available in real systems. However, the design of output-feedback controllers poses additional challenges in LMI-based methods due to convexity problems in acquiring the design conditions.

This paper provides a solution to the open problem of robust state-feedback and static output-feedback controllers design for discrete-time uncertain systems through offline design by RL methods. The controllers guarantee the stability and the pole placement of the closed-loop eigenvalues in a disk on the complex plane. The proposed technique is based on the use of the reinforcement learning approach by employing the SARSA (State-Action-Reward-State-Action) algorithm. The agent represents the controller, which is considered directly as a design variable in the method. This is an important feature of this method since there is no need for transformations to recover the gain matrices. Moreover, unlike existing methods, there are no concerns about specific structures for a Lyapunov function that allows obtaining convex conditions. The agent interacts with the plant model in an offline manner and, while interacting with it, learns the controller values that guarantee the desired pole placement. A time-varying incremental step is adopted to change the agent along the process. The method makes use of a switched subset for the uncertain parameters to reduce the computational burden. The technique starts with the evaluation of the vertices of the polytope, and after the desired state is reached by its vertices, a grid on the uncertain parameters is performed. If the designed controller shows satisfactory performance through the grid, a hard stability certificate is obtained via an LMI-based method, which is inspired by existing methods from the literature [18,29].

2. Problem statement

In this section, the considered system characteristics and uncertainties will be defined, as well as the utilized control laws. An LMI-based method to certify stability will be stated followed by a general exposition of the RL problem and its elements.

2.1. The uncertain system

Consider the following Linear Time-Invariant (LTI) discrete-time uncertain system

$$\begin{aligned} x(k+1) &= A(\alpha)x(k) + B(\alpha)u(k), \\ y(k) &= C(\alpha)x(k), \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^n$ is the state vector, $u \in \mathbb{R}^{n_u}$ is the control input, $y \in \mathbb{R}^{n_y}$ is the output vector and k is the time instant. The matrices $A(\alpha) \in \mathbb{R}^{n\times n}, B(\alpha) \in \mathbb{R}^{n\times n_u}$, and $C(\alpha) \in \mathbb{R}^{n_y\times n}$ belong to a polytopic domain, whose parameters are time-invariant and are included in the vector α. A generic matrix $G(\alpha)$ is given by:

$$\begin{aligned} &G(\alpha) = \textstyle\sum_{v=1}^{V}\alpha_v G_v, \quad \alpha \in \Lambda_V, \\ &\Lambda_V = \{\alpha \in \mathbb{R}^V : \textstyle\sum_{v=1}^{V}\alpha_v = 1; \alpha_v \geqslant 0, v = 1, \ldots, V\}, \end{aligned} \tag{2}$$

where V is the number of vertices of the polytope. The vertices of the polytope are known and can be used to define the region in which the uncertainties lie.

Our goal is to design robust state-feedback controllers, consisting in the gains $K_x \in \mathbb{R}^{n_u \times n}$ and the following control law:

$$u(k) = K_x x(k), \tag{3}$$

and static output-feedback controllers, consisting in the matrix gain $K_y \in \mathbb{R}^{n_u \times n_y}$, and the following law:

$$u(k) = K_y y(k). \tag{4}$$

Both state-feedback and static output-feedback control problems for uncertain systems are well-known challenging issues, that remain without an optimal solution. Even though several LMI-based methods have been reported in the last years, the conservatism introduced by the choice of a specific Lyapunov function, and the use of technical lemmas to provide convex conditions, still are problems to be overcome.

In this paper, the controllers will be designed to guarantee that the closed-loop poles lie in the disk $D(\sigma, \rho)$ with center σ and radius ρ. The closed-loop system for the state-feedback control action is obtained by using the control input (3) in the system (1), which gives

$$\begin{aligned} x(k+1) &= (A(\alpha) + B(\alpha)K_x)x(k), \\ &= A_{cl}(\alpha)x(k). \end{aligned} \tag{5}$$

The closed-loop system for static output-feedback control is attained by plugging the control action (4) in the system (1), yielding

$$\begin{aligned} x(k+1) &= \big(A(\alpha) + B(\alpha)K_y C(\alpha)\big)x(k), \\ &= A_{cl}(\alpha)x(k). \end{aligned} \tag{6}$$

To design the gain matrices K_x, and K_y, we propose a new method based on a machine learning technique, more specifically, in this work the reinforcement learning approach will be employed. Although it is known as a stochastic approach, in this paper a stability certificate for the closed-loop system will be obtained by means of a Lyapunov function.

2.2. Stability certificate

The matrix $A_{cl}(\alpha)$ in the closed-loop systems (5), and (6) must have all the eigenvalues in the unit circle to guarantee the stability of the system. To check the stability of the system, the following result, which is an analysis condition, will be employed.

Theorem 1. The system $x(k+1) = A_{cl}(\alpha)x(k)$ is asymptotically stable, if and only if there exist $N \geqslant 1$ and symmetric matrices $P_i(\alpha) \in \mathbb{R}^{n \times n}, i = 1, 2, \ldots, N$, such that the following conditions hold:

$$\sum_{i=j}^{N} P_i(\alpha) > 0, \quad j = 1, 2, \ldots, N, \quad \alpha \in \Lambda_V, \tag{7}$$

$$\sum_{i=1}^{N} A_{cl}(\alpha)^{i^T} P_i(\alpha) A_{cl}(\alpha)^i - P_i(\alpha) < 0, \quad \alpha \in \Lambda_V. \tag{8}$$

Proof. The proof is presented in A.

In what follows, a brief summary of the RL method is presented.

2.3. Fundaments of reinforcement learning

Fig. 1 presents the generic outline of an RL problem. In this schematic, it is possible to consider the agent as the controller and the environment as the plant and its respective characteristics. The problem, then, involves a loop where the agent takes an action a_t towards the environment and receives a reading of the consequential state s_{t+1} and a reward signal r_{t+1}. Through this new state, and mainly through the reward signal, the agent will learn how well his action was, given the previous state, and continue to interact with the environment until it reaches a terminal state. In the feedback control problem, the states can be defined as ranges of the closed-loop system poles values in the complex plane.

Problems in RL operate in a Markov Decision Process (MDP). The mathematical formulation of an MDP considers the existence of the following.

- The State-Space ($s \in \mathbb{S}$) is the set of states of the environment.
- The Action Space ($a \in \mathbb{A}$) contains the set of possible actions the agent may take in each iteration.
- The Reward Signal ($r \in \mathbb{R}$) is a scalar that evaluates the quality of the action taken in each iteration.
- The Policy ($\pi(a_t|s_t)$) chooses an action a_t given the current observed state s_t.

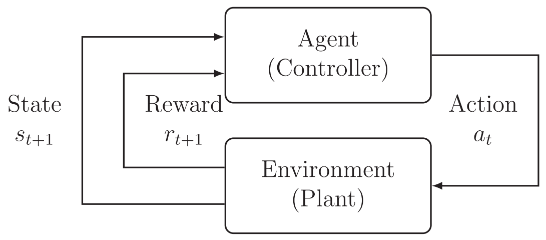

Fig. 1. Agent-Environment interaction in Reinforcement Learning. The agent interacts with the Environment through actions, receiving a reward evaluating its action quality and a reading of its new state.

- The Transition Probability Distribution ($\mathbb{P}(s_{t+1}, r_{t+1}|s_t, a_t)$) that describes the probability of a reward r_{t+1} being provided given an action a_t that translates the system from the state s_t to a state s_{t+1}.

The agent's overall objective is to maximize the accumulated discounted reward R_t. The use of the subscript t refers to the iteration number. This sum of rewards uses a discount factor $\gamma \in [0, 1]$ that weights how much the returns of short and long-term rewards shall be taken into account.

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{i=0}^{\infty} \gamma^k r_{t+i+1} \tag{9}$$

It is worth noting the existence of Model-Free methods [33], which do not rely on a Transition Probability Distribution to solve RL problems. This is particularly useful in the considered problem, as $\mathbb{P}$ is not known and the systems are deterministic in nature.

The use of Model-Free methods will impact the process in which the agent defines its policies. Simply put, the agent will choose a greedy policy, as it seeks to maximize the expected accumulated returns through its actions. By not utilizing any kind of transition probability, this may be done through the state-action function $Q(s, a)$. The value of this function defines which action is expected to be the worthiest to be taken, given the agent state s_t, a policy π and it is described by the following generic equation.

$$Q_\pi(s_t, a_t) = \mathbb{E}_\pi[R_t|s_t = s, a_t = a] \tag{10}$$

There are many techniques to evaluate the state-action function. Among them, it is worth citing Dynamic Programming (DP) methods [4], Monte Carlo (MC) based methods [15] and Temporal Difference (TD) methods [33]. Methods based on DP may be vulnerable to the called "curse of dimensionality", where the computational cost required to solve big state space problems becomes impracticable, and MC methods require obtaining the average of many sequences until an optimal policy emerges. In this paper, a method based on TD was selected, as the agent can learn from the environment by interacting with it, even in an offline manner, offering a decision-making process.

3. Feedback control using SARSA

The SARSA algorithm is a TD method based on the tuple $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$. Departing from the state s_t, the agent chooses an action a_t resulting in a transition to s_{t+1} and a reward signal r_{t+1}. Based on this new state, the agent then chooses a new action a_{t+1}. This series of decisions one iteration forward is then employed to update the state-action function $Q(s_t, a_t)$.

The state-action function (10) assumes a look-up table format in this instance, where the number of rows is equivalent to the dimension of the state space $\mathbb{S}$, and the number of columns is equal to the number of possible actions the agent is able to take. In the SARSA algorithm, the update rule to the state-action function is described by the following equation

$$Q(s_t, a_t) = Q(s_t, a_t) + \mu(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)), \tag{11}$$

where $\mu \in [0, 1]$ is a learning coefficient, and $\gamma \in [0, 1]$ a discount factor.

An important aspect of RL problems is the balancing between exploration and exploitation [8]. Choosing an exclusively greedy policy favors only the latter, preventing the agent from exploring actions that would be better in the long term. To try to prevent this, the ϵ-greedy strategy was selected to define the agent's decision criteria [33]. The a_t decision will be made according to the following rule, noting that a is the vector with all possible actions.

$$a_t = \begin{cases} \max\limits_{a_t} Q(s_t, a), & \text{with probability } 1 - \epsilon. \\ \text{random action}, & \text{with probability } \epsilon. \end{cases} \tag{12}$$

The iterative process of SARSA is an on-policy approach, thus, the policy chosen by the agent is the same evaluated in the learning process. As the utilized method is stochastic, to ensure the convergence of (11), a GLIE (Greedy in the Limit with Infinite Exploration) policy criteria was utilized. This criteria requires two properties: i) each action must be executed infinitely often in every state that is visited often; ii) in the limit, the learning policy is greedy with respect to the state-action function value with probability 1.

To ensure the greedy policy in the limit, the value of ϵ, as utilized in (12), will be dependent on the iteration number t and the maximum iteration number t_{max}, departing from a maximum value of ϵ_{max}.

$$\epsilon_t = \left(\frac{t_{max} - t}{t_{max}}\right)\epsilon_{max} \tag{13}$$

Furthermore, SARSA convergence is shown in Theorem 2, as proposed by [32]:

Theorem 2 [32]. Considering a finite state-action MDP and a GLIE learning policy π given as a set of probabilities $Pr(a|s, t, \eta_t(s), Q)$. a_t is the chosen action according to π at the iteration t, where π uses $Q = Q(s_t, a_t)$, where $Q(s_t, a_t)$ is computed by (11). Then $Q(s_t, a_t)$ converges to the optimal state-action function value and policy π_t converges to the optimal policy, as long as the conditions of immediate rewards, state transitions and learning rates satisfy the following:

1. The Q values are stored in a lookup table
2. The learning rate satisfy $0 \leqslant \mu \leqslant 1, \sum_t \mu = \infty$ and $\sum_t \mu^2 < \infty$.
3. $Var\{r(s, a)\} < \infty$

where, beyond the already declared variables, $\eta_t(s)$ is the number of times the s state was visited before iteration t and $Pr(a|s, t, \eta_t(s), Q)$ is the probability that action a is selected given the history.

Proof. The proof can be found in [32].

3.1. State space

The state space will be constructed on the complex plane. Departing from the disk $D(\sigma, \rho)$, various concentrical disks in σ will be considered, each having an increment in radius size of $\Delta\rho$, as illustrated in Fig. 2. From the Figure, we see the numbered regions $1, 2, 3, \ldots$, that are used to locate the poles of the closed-loop system. This means that based on the position of the eigenvalues in the complex plane, the agent's state takes on a different value and location.

At each iteration t the algorithm must compute the state of the agent. Define λ as the function that calculates the system eigenvalues and Θ_t as

$$\begin{aligned} \Theta_t &= \max|\lambda(A_{cl}(\alpha)) - \sigma|, \quad &\text{for } \alpha \in \mathscr{P}_s, \quad \text{for } t \geqslant 1, \\ \Theta_0 &= \max|\lambda(A(\alpha)) - \sigma|, \quad &\text{for } \alpha \in \mathscr{P}_s, \end{aligned} \tag{14}$$

where $\mathscr{P}_s \subset \Lambda_V$. Note that Θ_t is used to indicate the maximum distance of the eigenvalues of the closed-loop matrices to the center σ of the disk. To reduce the computational burden of the method, the subset $\mathscr{P}_s$ will be considered as follows

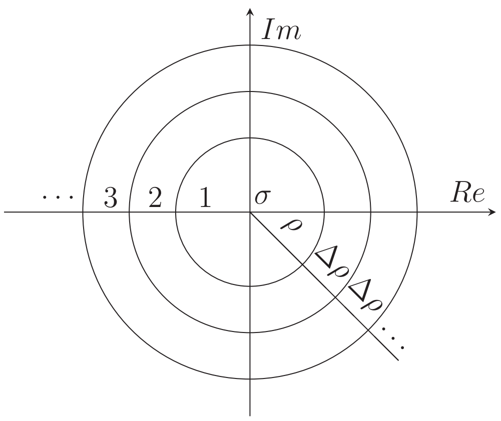

Fig. 2. Concentric circles for $\sigma = 0$ defining the utilized state space $\mathbb{S}$. The agent's current state location is defined by (14) in relation to the concentric circles.

$$\mathscr{P}_s = \begin{cases} \text{vertices of } A_{cl}(\alpha), & \text{if } \Theta_t \geqslant \rho, \\ \text{grid on } A_{cl}(\alpha), & \text{if } \Theta_t < \rho. \end{cases} \tag{15}$$

The procedure in (15) checks only the vertices of the closed-loop system when the agent is outside of the region of interest ($\Theta_t \geqslant \rho$). When all the vertices have poles in the desired position of the complex plane ($\Theta_t < \rho$), the set $\mathscr{P}_s$ switches to a grid on the uncertain parameter α where 1000 randomly generated points are considered.

Remark 1. It is important to highlight that the stability of the vertices of the uncertain polytopic system is only a necessary condition to certify the stability of the polytope. That is the main reasoning behind the idea of using a switching set $\mathscr{P}_s$.

For a given Θ_t, it will be possible to define the state of the system according to Fig. 2. In this way, when $\Theta_t < \rho$, and $\mathscr{P}_s$ is a grid on $A_{cl}(\alpha)$, the agent will be located in the terminal state, successfully performing the pole placement.

To avoid the creation of a greater number of states, the maximum radius is defined by considering $1.02\Theta_0$, where Θ_0 is computed using (14), under the (15) subset. If during the algorithm execution $\Theta_t > 1.02\Theta_0$, the agent will be positioned in the last state defined at the beginning of the algorithm.

3.2. Space of actions

Given that the agent represents the controller itself, the actions will consist of individual increments in the values of the gain K, being it either $K = K_x$ or $K = K_y$. These increments may be positive or negative, which results in $2nn_u$ possible actions for (3) and $2n_yn_u$ for (4).

To ease the convergence, mainly when (14) is close to the value of ρ, a time-varying incremental step is defined as $\Theta_t/1000$. When the distance Θ_t increases, the incremental step also increases. In the same way, when Θ_t is close to the terminal state, the incremental step will be smaller.

To better illustrate the Action Space and gain increment process, consider the control law (3) and $n = 2, n_u = 1$. We will have

$$\mathscr{A} = \left\{ \left[\tfrac{\Theta_t}{1000} \quad 0\right], \left[0 \quad \tfrac{\Theta_t}{1000}\right], \left[-\tfrac{\Theta_t}{1000} \quad 0\right], \left[0 \quad -\tfrac{\Theta_t}{1000}\right] \right\}, \tag{16}$$

and $K = K_x = [K_1 K_2]$. The increment is done by directly adding the chosen action a_t to K. Note that the set of actions $\mathscr{A}$ could be modified to include or to remove actions.

3.3. Reward process and algorithm

Next, the reward process will be defined. Its objective is to encourage the agent to move towards the terminal state, that is, towards state 1, defined by the interior of the disk $D(\sigma, \rho)$. s_{t+1}, s_t refer to the number of the state, as seen in Fig. 2. Decreasing the value of Θ_t ($s_{t+1} < s_t$) results in a less negative reward, which becomes proportionally even less negative according to the number of states traversed towards the terminal state. Stagnating in the same state ($s_{t+1} = s_t$) is somewhat undesirable, receiving a more negative reward. Providing poorer performance ($s_{t+1} > s_t$) is rewarded with the most negative reward, which is also multiplied by the number of states traversed in the opposite direction of the terminal state, further discouraging the action taken. Ultimately, the reward process is described by the following function,

$$r_{t+1} = \begin{cases} -10(s_{t+1} - s_t), & s_{t+1} > s_t, \\ -5, & s_{t+1} = s_t, \\ \frac{-1}{(s_t - s_{t+1})}, & s_{t+1} < s_t, \end{cases} \tag{17}$$

Having defined the outline of the RL problem, the SARSA algorithm implemented is the following.

Algorithm 1:SARSA Algorithm

```
SARSA Algorithm
1  Input $\sigma$, $\rho$, $A(\alpha), B(\alpha), C(\alpha)$
2  Initialize the parameters $\mu$, $\gamma$, $\epsilon_{max}$
3  Initialize $K = 0$
4  Compute $\Theta_0$ using (14) with $\mathscr{P}_s$ as in (15)
5  Initialize $\Delta\rho$
6  Define the maximum radius through $1.02\Theta_0$
7  Initialize $Q(s,a) = 0$
8  Identify $s_t$ with $\Theta_0$
9  Choose $a_t \in \mathscr{A}$ using (12) considering $\Theta_0$ and $\epsilon_0$ as in (13)
10 While $t \leq t_{max}$
11   $K = K + a_t$
12   Compute $\Theta_{t+1}$ with $K$ using (14) with $\mathscr{P}_s$ as in (15)
13   Identify $s_{t+1}$ with $\Theta_{t+1}$ and the reward $r_{t+1}$ as in (17)
14   Choose $a_{t+1}$ using (12), considering $\Theta_{t+1}$ and $\epsilon_t$ as in (13)
15   Update $Q(s_t, a_t)$ using (11)
16   $s_t = s_{t+1}, \;\; a_t = a_{t+1}, \;\; \Theta_t = \Theta_{t+1}$
17   $t = t + 1$
18   If $(s_t = 1)$ then
19     If Theorem 1 is feasible considering $K$ then
20       Output $K$
21       Break
22 End While
```

Remark 2. The algorithm has the state-space model matrices and the desired ρ and σ for the pole-placement as inputs. If the process converges, the gain matrix K_x or K_y that stabilize the closed-loop system is the output.

Remark 3. Even though the iterative process is bounded by a maximum number of iterations t_{max}, the process will terminate as soon as Theorem 1 is feasible to a given K.

Remark 4. Note that the initial condition considered for the gain matrices K_x and K_y is simply a matrix of zeros. However, different initializations could be considered, leading to different results.

Remark 5. The last step performed by the algorithm is used to provide a hard stability certificate for the closed-loop system via Theorem 1. Even though the gain matrices are directly the design variables in the algorithm, and the search for stabilizing gains does not rely on a Lyapunov function, this step is employed to provide guarantees that the closed-loop system is asymptotically stable. Different from the control design problem based on LMI methods, the stability conditions for uncertain polytopic systems via LMI provide necessary and sufficient conditions to certificate the stability of the closed-loop system.

Remark 6. The values of σ and ρ may be changed to find different solutions, being $\sigma = 0, \rho = 1$ the most conservative scenario in terms of performance. Also, not every pole placement is possible to every system, which makes the decision of these parameter values important to the method convergence.

4. Numerical experiments

Benchmark examples borrowed from the literature were selected to test the method and to illustrate its effectiveness. As far as the authors' knowledge, there are no RL methods in the literature dealing with state-feedback and static output-feedback control design for uncertain polytopic systems. For this reason, we focus on comparisons with LMI-based methods. These examples include the system utilized in [10], where a parameter can be increased as a way to test a method's conservatism, as well as a set of randomly generated systems proposed in [23]. The numerical experiments were performed using MATLAB (R2016b) 64 bits for Windows 10, with the aid of the parsers YALMIP [20] and ROLMIP [1], and the solver MOSEK [2], in a machine with Intel Core i7-8550U (1.8 GHz) processor and 12 GB RAM. In all experiments the following parameters have been employed for the proposed method: $\gamma = 0.9, \mu = 0.3, \epsilon_{max} = 0.3$, and $\Delta\rho = \frac{\Theta_0}{2000}$ as computed by (14), and a maximum number of iterations $t_{max} = 500000$.

4.1. Example 1

Consider the discrete-time system borrowed from [10], with uncertain matrices given by

$$A = \begin{bmatrix} 0.8 & -0.25 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0.2 & 0.03 \\ 0 & 0 & 1 & 0 \end{bmatrix} + \upsilon \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0.8 \\ -0.5 \\ 0 \\ 1 \end{bmatrix}^T,$$
$$B = \beta \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + (1-\beta) \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad 0 \leqslant \beta \leqslant 1,$$

where $|\upsilon| < \tau$. The system can be described by a polytope with $V = 4$ vertices. The goal is to determine the maximum value τ such that the system can be stabilized by a state-feedback control law as in (5). The method in [10] is able to design controllers that stabilize the system up until $\tau = 0.8892$, while the method proposed in this paper achieves $\tau = 1.2400$, an improvement of more than 39.45%.

Since the method is stochastic in nature and provides different answers after each iterative process, the algorithm was executed 100 times to test its capability to reach the terminal state. In 99% of the time, the method was able to provide gains which made the closed-loop system (3) meet the project criteria. For this example, it was considered $\rho = 1$ and $\sigma = 0$. One of the resulting gains was $K = [0.3547 \quad -0.2596 \quad -1.2610 \quad -0.2477]$, and required 3.9814 seconds of computational time. Given the obtained state-feedback controller, the stability certificate proposed by Theorem 1 was able to guarantee closed-loop stability for $N = 2$.

In Fig. 3, the spectrum of the closed-loop system $A_{cl}(\alpha)$, as in (5), is plotted for different stages of the convergence process. As it can be seen, the method is able to place all the poles in the unit circle throughout the execution of the algorithm.

4.2. Example 2

In this example we intend to perform a statistic comparison with the approaches presented in [10,23]. To this end, a database proposed in [23] is considered. This database is composed of systems that are open-loop unstable and cannot be stabilized by quadratically stabilizing feedback gains.

Uncertain polytopic systems of dimensions $n_u = 1, n = 2, 3, 4, 5$, and vertices $V = 2, 3, 4, 5$, were considered. The percentage of systems that were stabilized by 5 executions of the RL-based method was compared to the method SAL (Stabilizability with Affine Lyapunov matrix) [10] and the method in [23] that rely on the use of a scalar parameter search. Moreover, the approach in [30], where a scalar parameter search is combined with a heuristic method through an algorithm also was employed for comparisons. It is worth noting that for the state-feedback case the method presented in [30] reports only the values for dimensions $n = 2$, and $n = 3$, and for the number of vertices from $V = 2$ to $V = 5$. The results are presented in Table 1.

It can be seen that the RL-based method provides competitive results when compared to the different techniques available in the literature. In this example, the poles were placed in the unit circle centered in zero. As we have mentioned before, the parameters used in the RL algorithm are the same for all tests. However, it is possible to adjust the parameters $\gamma, \mu, \epsilon_{max}$, and $\Delta\rho$ to treat different classes of problems.

4.3. Example 3

To illustrate the pole placement capabilities of the method, consider the randomly generated discrete-time system with $n = 2, n_u = 1$, and $V = 5$ vertices, which was created following the procedures proposed by [23], and whose matrices are given by:

$$A_1 = \begin{bmatrix} 6.7130 & 0.6065 \\ -17.0763 & -0.1702 \end{bmatrix}, A_2 = \begin{bmatrix} -15.6627 & -0.8770 \\ -15.6627 & 0.3010 \end{bmatrix},$$

$$A_3 = \begin{bmatrix} -26.2615 & -1.3832 \\ -5.2994 & -0.3709 \end{bmatrix}, A_4 = \begin{bmatrix} -20.9621 & -0.5411 \\ -26.2615 & -1.3832 \end{bmatrix},$$

$$A_5 = \begin{bmatrix} 26.4971 & 1.1476 \\ 10.3633 & 0.2706 \end{bmatrix}, B_1 = \begin{bmatrix} -1 \\ 3 \end{bmatrix}, B_2 = \begin{bmatrix} 3 \\ 3 \end{bmatrix},$$

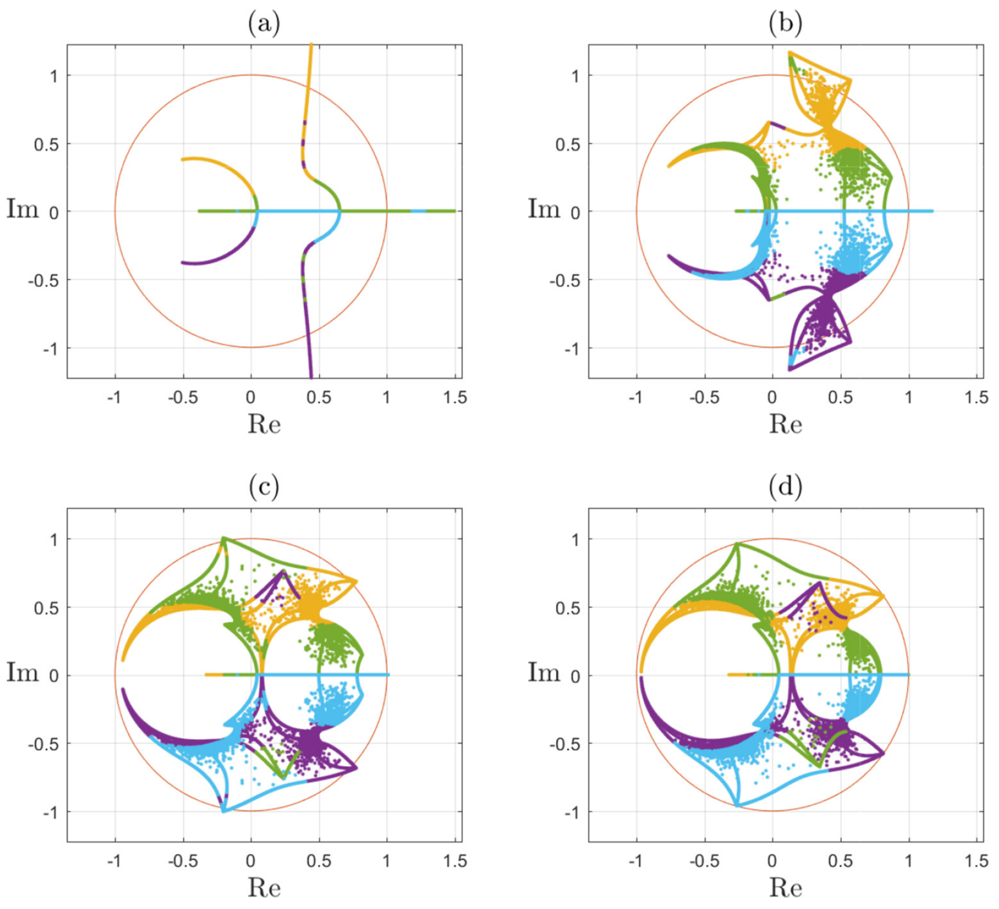

Fig. 3. Spectrum of the closed-loop matrix taken from different stages of the convergence process. (a) is the initial time step ($K_x = [0 \quad 0 \quad 0 \quad 0]$ and $\Theta = 1.4912$). (b) is ($K_x = [-0.1475 \quad -0.1815 \quad -0.7789 \quad -0.3312]$ and $\Theta = 1.1734$). (c) is ($K_x = [0.1627 \quad -0.2279 \quad -1.2467 \quad -0.3636]$ and $\Theta = 1.0328$) (d) is the end of the iterative process ($K_x = [0.3547 \quad -0.2596 \quad -1.2610 \quad -0.2477]$ and $\Theta = 0.9998$).

Table 1
LTI systems stabilized by state-feedback controllers.

n	*V*	[10](%)	[23, Cor. 1](%)	[30](%)	RL(%)
2	2	80.0	85.0	91.0	95.6
	3	90.0	93.0	95.0	96.2
	4	91.0	91.0	96.0	90.6
	5	89.0	93.0	99.0	93.6
3	2	85.0	86.0	92.0	98.4
	3	97.0	97.0	100.0	99.2
	4	95.0	95.0	96.0	99.4
	5	92.0	96.0	99.0	99.0
4	2	89.0	90.0	–	95.0
	3	94.0	94.0	–	99.0
	4	96.0	96.0	–	99.4
	5	94.0	94.0	–	99.6
5	2	94.0	95.0	–	83.8
	3	96.0	96.0	–	94.0
	4	93.0	93.0	–	97.2
	5	93.0	95.0	–	98.4
Total		91.8	93.1	–	96.2

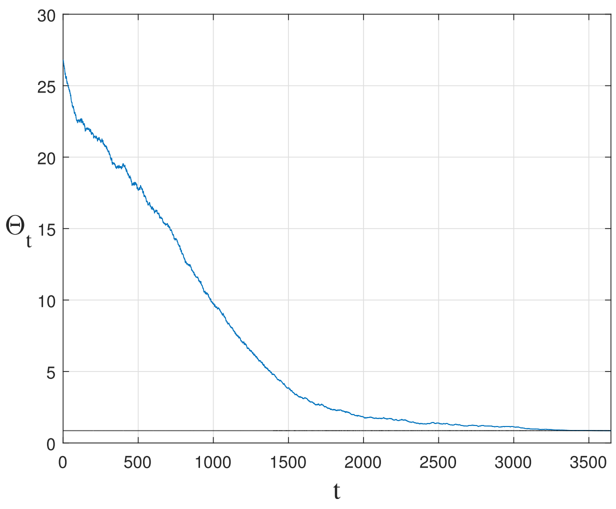

Fig. 4. Convergence process for the mean Θ_t of 100 algorithm executions considering Example 3 along the iterations of the algorithm. (−−) is the value of ρ.

$$B_3 = \begin{bmatrix} 5 \\ 1 \end{bmatrix}, B_4 = \begin{bmatrix} 4 \\ 5 \end{bmatrix}, B_5 = \begin{bmatrix} -5 \\ -2 \end{bmatrix}, C = [\,5.2994 \quad 0.1353\,].$$

The goal is to design a static output-feedback controller that places all the poles of the closed-loop system in the disk of center $\sigma = 0.1$, with the smallest possible radius ρ. This value was found to be $\rho = 0.86$. The proposed SARSA-based algorithm was executed 100 times, and was able to stabilize and perform the pole placement of the closed-loop system (4) in every execution.

One of these iterative processes resulted in the gain $K_y = 0.9994$, requiring 2.2221 seconds of computational time. The convergence process for Θ_t, which represents the distance of the poles to the center of the disk $D(0.1, 0.86)$, is depicted in Fig. 4, which contains the mean answer of the 100 algorithm executions. It is possible to see that the convergence process is not monotonically decreasing, however, along with the iterations of the algorithm the agent can achieve the desired goal, which is placing all the poles of the closed-loop system in the defined disk. The spectrum of the closed-loop system is shown in Fig. 5. The method in Theorem 1 $(N = 1)$ was employed to certify the stability of the closed-loop.

The iterative LMI-based technique proposed by [28], to design memory static output-feedback controllers, was not able to find a solution for this example, even when there were no constraints for pole placement. This method considers a buffer with the past output signals of the system in its control law, being a more complex method to design and implement. The iterative memory method has been tested up to $y(k-7)$ (seven memories) seeking to design a memory controller, and it did not present feasible solutions.

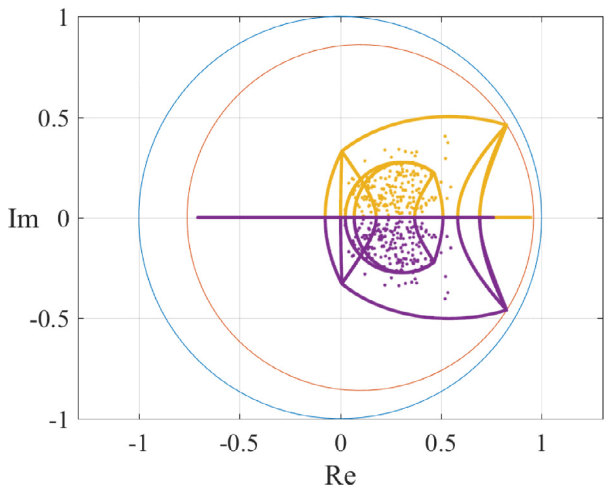

Fig. 5. Spectrum of the closed-loop matrix. (—) is the unit circle and (—) is the disk $D(0.1, 0.86)$. In the terminal state, the resulting gain $K_y = 0.9994$ provided $\Theta = 0.8590$.

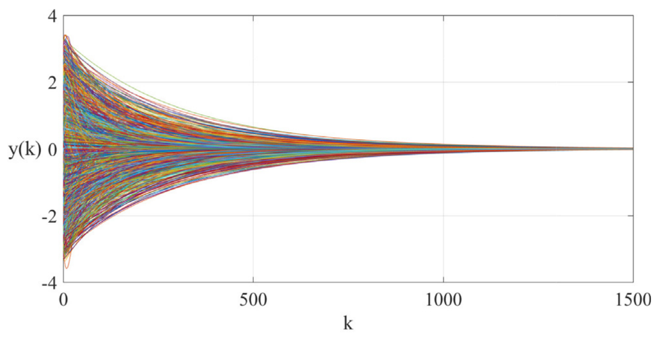

Fig. 6. Time-based simulation of the APS under 1000 randomly generated initial conditions x_0 and α. $K_y = -0.3806$.

4.4. Example 4

To apply the proposed technique in a real scenario, an Angular Position System (APS) is borrowed and adapted from [34]. It consists of a system with an antenna that moves towards a flying target through the use of a motor. The discretized state-space model with sample time of $0.1s$ is as follows:

$$A(\alpha) = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 - 0.1\delta \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0.1\kappa \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

where $0.1s^{-1} \leqslant \delta \leqslant 10s^{-1}$ is proportional to the viscous friction coefficient of the rotation parts of the system, $\kappa = 0.787rad^{-1}V^{-1}s^{-2}$.

The goal in this example is to design a static output-feedback controller that is applicable in a real-world scenario. The pole-placement parameters were set in $\rho = 0.9970, \sigma = 0$. The algorithm was executed 100 times, and in all of them, the technique was successful.

Randomly selecting one of the designed controllers, $K_y = -0.3806$, whose closed-loop stability was certified by Theorem 1 ($N = 1$). In order to test the system response in multiple scenarios, 1000 different combinations of initial conditions $x(0)$ and time-invariant parameter α are randomly selected. In this case, $(-\pi, -2) \leqslant x_0 \leqslant (\pi, 2)$ and α_1 and α_2 belong to the unit simplex presented in (2).

Fig. 6 shows the designed controller was able to stabilize the system even when it is subjected to different initial conditions and parameter uncertainties. This illustrates the robustness of the method in front of polytopic uncertainties in the model.

5. Conclusions

The design of state-feedback and output-feedback controllers poses a problem with many possible solutions, which are, however, susceptible to a certain conservatism. Methods which grant a stability certificate, however, are less prone to said conservatism and provide necessary and sufficient conditions to guarantee uncertain systems stability. This paper exploits this idea, and proposed a solution to the problem of robust state-feedback and static output-feedback controllers design for discrete-time uncertain systems with pole placement criteria, which proved itself less conservative than other methods found in the literature, and whose system characteristics are not as well explored by RL-based methods. The reinforcement learning technique was employed allowing the controller to be used directly as a design variable. The efficacy of the approach was demonstrated through benchmark examples from the literature, and exhaustive testing to prove the method's robustness even when given its stochastic nature, illustrating its capabilities in providing stabilizing controllers for uncertain systems. A real scenario example with an Angular Position System considering multiple values for the initial conditions and uncertainty parameter has shown the designed controller's effectiveness. Future directions include the use of performance criteria such as the $\mathscr{H}_2$ and the $\mathscr{H}_\infty$ problems and the use of deep reinforcement learning techniques.

CRediT authorship contribution statement

Pedro M. Oliveira: Writing - original draft, Conceptualization, Methodology, Software, Visualization, Investigation, Validation. **Jonathan M. Palma:** Writing - original draft, Conceptualization, Methodology, Visualization, Investigation, Supervision, Validation, Writing - review & editing, Funding acquisition. **Erivelton G. Nepomuceno:** Conceptualization, Methodology, Visualization, Investigation, Supervision, Validation, Writing - review & editing. **Márcio J. Lacerda:** Writing

[1] To shorten the notation we employ $x_k = x(k)$ in the Appendix.

- original draft, Conceptualization, Methodology, Visualization, Investigation, Supervision, Validation, Writing - review & editing, Funding acquisition, Project administration.

Declaration of Competing Interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Marcio J. Lacerda reports financial support was provided by Minas Gerais State Foundation of Support to the Research. Marcio J. Lacerda reports financial support was provided by National Council for Scientific and Technological Development. Pedro M. Oliveira reports was provided by Minas Gerais State Foundation of Support to the Research.

Acknowledgements

This work was supported in part by the Brazilian agencies CNPq, (grant 315538/2021-0), the Minas Gerais State Research Foundation (FAPEMIG) (grant PAPG-13121), and the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001, and Chilean agencies ANID-fondecyt de iniciación grant 11201049.

Appendix A. Proof of Theorem 1

By multiplying (8) by $x_k^\top$ on the left and by x_k[1] on the right yields

$$\begin{aligned} &x_k^\top A_{cl}(\alpha)^\top P_1(\alpha)A_{cl}(\alpha)x_k - x_k^\top P_1(\alpha)x_k + x_k^\top A_{cl}(\alpha)^{2^\top} P_2(\alpha)A_{cl}(\alpha)^2 x_k \\ &-x_k^\top P_2(\alpha)x_k + \ldots + x_k^\top A_{cl}(\alpha)^{N^\top} P_N(\alpha)A_{cl}(\alpha)^N x_k - x_k^\top P_N(\alpha)x_k < 0. \end{aligned} \tag{A.1}$$

Define $V_i(x_k) = x_k^\top P_i(\alpha)x_k, i = 1, \ldots, N$, and rewrite the inequality (A.1) as

$$V_1(x_{k+1}) + V_2(x_{k+2}) + \ldots + V_N(x_{k+N}) < V_1(x_k) + V_2(x_k) + \ldots + V_N(x_k).$$

Adding the term $S = \sum_{j=2}^{N}\sum_{i=j}^{N} V_i(x_{k+j-1})$ to both sides it follows that

$$V_1(x_{k+1}) + V_2(x_{k+2}) + \ldots + V_N(x_{k+N}) + S < V_1(x_k) + V_2(x_k) + \ldots + V_N(x_k) + S. \tag{A.2}$$

In this way, one can rewrite (A.2) as $W(x_{k+1}) - W(x_k) < 0$, with $W(x_k) = \sum_{j=1}^{N}\sum_{i=j}^{N} V_i(x_{k+j-1})$. Moreover, $W(x_k)$ can be written as

$$\begin{aligned} W(x_k) = {} & x_k^\top (P_1(\alpha) + P_2(\alpha) + \ldots + P_N(\alpha))x_k + x_{k+1}^\top (P_2(\alpha) + \ldots + P_N(\alpha))x_{k+1} + \ldots \\ &+ x_{k+N-2}^\top (P_{N-1}(\alpha) + P_N(\alpha))x_{k+N-2} + x_{k+N-1}^\top P_N(\alpha)x_{k+N-1}. \end{aligned} \tag{A.3}$$

Note that if the conditions (7) hold then the Lyapunov function (A.3) is positive definite. From (A.2) one concluded that $W(x_{k+1}) - W(x_k) < 0$. Thus, $W(x_k)$ is a Lyapunov function that assures the asymptotic stability of the closed-loop system $x(k+1) = A_{cl}(\alpha)x(k)$.

Necessity: Assume that the closed-loop system $x(k+1) = A_{cl}(\alpha)x(k)$ is asymptotically stable. Considering Theorem 1 with matrices $P_i(\alpha) = 0, i = 1, \ldots, N-1$ and $P_N(\alpha) = P$ a constant matrix. In this way condition (8) yields

$$A_{cl}(\alpha)^{N^\top} P A_{cl}(\alpha)^N - P < 0. \tag{A.4}$$

Notice that the asymptotic stability assumption guarantees that a N value exists and satisfies condition (A.4) with $P_N(\alpha) = P > 0$, as stated in (7).

References

[1] C.M. Agulhari, A. Felipe, R.C.L.F. Oliveira, P.L.D. Peres, Algorithm 998: The Robust LMI Parser - A Toolbox to Construct LMI Conditions for Uncertain Systems, ACM Trans. Math. Software 45 (3) (2019) 36:1–36:25.

[2] E.D. Andersen, K.D. Andersen, The MOSEK interior point optimizer for linear programming: An implementation of the homogeneous algorithm, in: H. Frenk, K. Roos, T. Terlaky, S. Zhang (Eds.), High Performance Optimization, Appl. Optimiz., vol. 33, US, Springer, 2000, pp. 197–232.

[3] K. Arulkumaran, M.P. Deisenroth, M. Brundage, A.A. Bharath, Deep reinforcement learning: A brief survey, IEEE Signal Process. Mag. 34 (6) (2017) 26–38.

[4] A.G. Barto, Reinforcement learning and dynamic programming, in: Analysis, Design and Evaluation of Man-Machine Systems 1995, Elsevier, 1995, pp. 407–412.

[5] Bosche, J., Bachelier, O., and Mehdi, D. (2004). Robust pole placement by static output feedback. In Proceedings of the 43rd IEEE Conference on Decision and Control, pages 869–874, Paradise Island, Bahamas.

[6] A.L. Carvalho Ottoni, E. Geraldo Nepomuceno, M. Santos de Oliveira, Development of a Pedagogical Graphical Interface for the Reinforcement Learning, IEEE Latin Am. Trans. 18 (01) (2020) 92–101.

[7] G. Chesi, LMI techniques for optimization over polynomials in control: A survey, IEEE Trans. Autom. Control 55 (11) (2010) 2500–2510.

[8] Coggan, M. (2004). Exploration and exploitation in reinforcement learning. Research supervised by Prof. Doina Precup, CRA-W DMP Project at McGill University.

[1] To shorten the notation we employ $x_k = x(k)$ in the Appendix.

[9] W. Cui, B. Zhang, Lyapunov-regularized reinforcement learning for power system transient stability, IEEE Control Syst. Lett. 6 (2022) 974–979.
[10] M.C. de Oliveira, J. Bernussou, J.C. Geromel, A new discrete-time robust stability condition, Syst. Control Lett. 37 (4) (1999) 261–265.
[11] J. Dong, G.-H. Yang, Static output feedback control synthesis for linear systems with time-invariant parametric uncertainties, IEEE Trans. Autom. Control 52 (10) (2007) 1930–1936.
[12] G. Garcia, J. Bernussou, Pole assignment for uncertain systems in a specified disk by state-feedback, IEEE Trans. Autom. Control 40 (1) (1995) 184–190.
[13] G. Garcia, J. Daafouz, J. Bernussou, Output feedback disk pole assignment for systems with positive real uncertainty, IEEE Trans. Autom. Control 41 (9) (1996) 1385–1391.
[14] I. Grondman, L. Busoniu, G.A. Lopes, R. Babuska, A survey of actor-critic reinforcement learning: Standard and natural policy gradients, IEEE Trans. Syst., Man, Cybern., Part C (Appl. Rev.) 42 (6) (2012) 1291–1307.
[15] T. Jaakkola, S.P. Singh, M.I. Jordan, Reinforcement learning algorithm for partially observable Markov decision problems, Adv. Neural Inform. Process. Syst. (1995) 345–352.
[16] B.R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A.A.A. Sallab, S. Yogamani, P. Pérez, Deep reinforcement learning for autonomous driving: A survey, IEEE Trans. Intell. Transp. Syst. (2021) 1–18.
[17] B. Kiumarsi, F.L. Lewis, Z.P. Jiang, $\mathscr{H}_\infty$ control of linear discrete-time systems: Off-policy reinforcement learning, Automatica 78 (2017) 144–152.
[18] M.J. Lacerda, P. Seiler, Stability of uncertain systems using Lyapunov functions with non-monotonic terms, Automatica 82 (2017) 187–193.
[19] Y.-P. Lin, X.-Y. Li, Reinforcement learning based on local state feature learning and policy adjustment, Inf. Sci. 154 (1–2) (2003) 59–70.
[20] Löfberg, J. (2004). YALMIP: A toolbox for modeling and optimization in MATLAB. In Proceedings of the 2004 IEEE International Symposium on Computer Aided Control Systems Design, pages 284–289, Taipei, Taiwan.
[21] N.C. Luong, D.T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, D.I. Kim, Applications of deep reinforcement learning in communications and networking: A survey, IEEE Commun. Surv. Tutor. 21 (4) (2019) 3133–3174.
[22] M. Mohammadi, M.M. Arefi, P. Setoodeh, O. Kaynak, Optimal tracking control based on reinforcement learning value iteration algorithm for time-delayed nonlinear systems with external disturbances and input constraints, Inf. Sci. 554 (2021) 84–98.
[23] Morais, C.F., Braga, M.F., Oliveira, R.C.L.F., and Peres, P.L.D. (2013). Robust state feedback control for discrete-time linear systems via LMIs with a scalar parameter. In Proceedings of the 2013 American Control Conference, pages 3876–3881, Washington, DC, USA.
[24] S. Mukherjee, T.L. Vu, On distributed model-free reinforcement learning control with stability guarantee, IEEE Control Syst. Lett. 5 (5) (2021) 1615–1620.
[25] A.L.C. Ottoni, E.G. Nepomuceno, M.S. de Oliveira, D.C.R. de Oliveira, Tuning of reinforcement learning parameters applied to SOP using the Scott-Knott method, Soft. Comput. 24 (6) (2020) 4441–4453.
[26] A.L.C. Ottoni, E.G. Nepomuceno, M.S. de Oliveira, D.C.R. de Oliveira, Reinforcement learning for the traveling salesman problem with refueling, Complex & Intelligent Systems, 2021.
[27] A.L.C. Ottoni, E.G. Nepomuceno, M.S. Oliveira, A Response Surface Model Approach to Parameter Estimation of Reinforcement Learning for the Travelling Salesman Problem, J. Control, Autom. Electr. Syst. (2018).
[28] J.M. Palma, C.F. Morais, R.C.L.F. Oliveira, Linear matrix inequality-based solution for memory static output-feedback control of discrete-time linear systems affected by time-varying parameters, Int. J. Robust Nonlinear Control 31 (9) (2021) 4324–4336.
[29] P.S.P. Pessim, V.J.S. Leite, M.J. Lacerda, Robust performance for uncertain systems via Lyapunov functions with higher order terms, J. Franklin Inst. 356 (5) (2019) 3139–3156.
[30] T.E. Rosa, C.F. Morais, R.C.L.F. Oliveira, New robust LMI synthesis conditions for mixed $\mathscr{H}_2/\mathscr{H}_\infty$ gain-scheduled reduced-order DOF control of discrete-time LPV systems, Int. J. Robust Nonlinear Control 28 (18) (2018) 6122–6145.
[31] M.S. Sadabadi, D. Peaucelle, From static output feedback to structured robust static output feedback: A survey, Annu. Rev. Control 42 (2016) 11–26.
[32] S. Singh, T. Jaakkola, M.L. Littman, C. Szepesvári, Convergence results for single-step on-policy reinforcement-learning algorithms, Mach. Learn. 38 (3) (2000) 287–308.
[33] R.S. Sutton, A.G. Barto, Reinforcement learning: An introduction, 2nd edition., MIT press, Cambridge, 2018.
[34] M. Wang, B. Xu, Observer-based guaranteed cost control of cyber-physical systems under dos jamming attacks, Eur. J. Control 48 (2019) 21–29.
[35] Yu, L., Han, Q.-L., and He, X.-X. (2003). Design of robust output feedback controllers with variance and disc closed-loop pole constraints. In Proceedings of the 42nd IEEE Conference on Decision and Control, pages 774–779, Maui, Hawaii USA.
[36] I.A. Zamfirache, R.-E. Precup, R.-C. Roman, E.M. Petriu, Policy iteration reinforcement learning-based control using a grey wolf optimizer algorithm, Inf. Sci. (2021).
[37] I.A. Zamfirache, R.-E. Precup, R.-C. Roman, E.M. Petriu, Reinforcement learning-based control using q-learning and gravitational search algorithm with experimental validation on a nonlinear servo system, Inf. Sci. 583 (2022) 99–120.